ZKHackathon ZKP circuit track

We welcome members of the community to collaborate with us in advancing the development of efficient circuits and R1CSs, as well as ZKP protocols for critical computations in blockchain applications. It is a critical step to enhance the efficiency and scalability of ZKP schemes for deployment in various applications, including zkRollups, zkBridges, and DeFi. We are confident that with the collective efforts of the community, we can achieve this goal and make contributions to the advancement of ZKP and blockchain technology.

Introduction

ZKP techniques have been widely adopted in many blockchain applications to improve the scalability and privacy. One popular use of ZKPs is in the implementation of zkRollups, which allow for the aggregation of many transactions into a single proof, thereby reducing the amount of data that needs to be processed and verified on the blockchain. Another important application is in privacy-preserving cryptocurrencies, where ZKPs are used to conceal the details of transactions, such as the sender, recipient, and amount involved. Despite their prominent advancement, the prover time of the ZKPs is still the bottleneck in many of these applications.

These applications require many common computations, including computing hash functions, verifying Merkle tree paths, and validating digital signatures. This track provides a unique opportunity to advance the frontiers of ZKP technology and create optimized circuits and protocols for these computations. The outcomes of this track will have a significant positive impact on both current ZKP deployments and the future development of ZKP applications, by increasing the efficiency of the schemes. Furthermore, the designs developed during this track will be open-source and have the potential to become standard primitives in future ZKP applications.

Program Task Description

Category 1: Circuits/R1CSs for cryptographic primitives

Description: cryptographic primitives such as hash functions and digital signatures are necessary building blocks in almost all ZKP applications on blockchain. The goal of this task is to design optimized circuits/R1CSs that have smaller sizes than existing ones compiled automatically by ZKP libraries.

Designated Tasks:

Designated Task 1.1: SHA-256
Designated Task 1.2: Keccak-256
Designated Task 1.3: BLS signature
Designated Task 1.4: ECDSA signature
Designated Task 1.5: Ed25519 signature

Category 2: Circuits/R1CSs for recursive SNARKs

Description: recursion/proof composition is a technique to verify the proof of a first SNARK scheme using a second SNARK. It is an important technique to reduce the proof size, to improve the total prover time and to aggregate multiple proofs for different computations. The goal of this task is to design optimized circuits/R1CSs for common primitives used in recursive SNARKs.

Designated Task 2.1: Cycles of elliptic curves: BLS12-377 to BW6-761

Recursive SNARKs such as Zexe and VeriZexe rely on a "two-chain" of elliptic curves to support one level of recursion. The objective of this task is to optimize the circuit/R1CS in order to enable such recursion on the two curves. Specifically, the inner curve is BLS12-377, which includes the embedded curve of Jubjub377 with a base field that matches the scalar field of BLS12-377. Meanwhile, the outer curve is BW6-761, which has BLS12-377 as an embedded curve. The goal is to implement the bilinear pairing of the inner curve as a circuit/R1CS on the outer curve. Participants can use existing libraries such as artworks to develop the circuits.

Designated Task 2.2: Verification of a Stark proof

Another type of recursive ZKPs is based on Stark. he objective is to enhance the circuit/R1CS of the Stark verification process, such that it can be combined with a second ZKP to minimize the size of the overall proof. To accomplish this, the task involves leveraging various building blocks, such as Merkle tree verification and hash functions that serve as random oracles.

Designated Task 2.3: Cross-field arithmetic: implementing additions and multiplications of one prime field over another prime.

It is often necessary to incorporate arithmetic operations for a different field within the local field supported by a SNARK or other cryptographic application. This task involves selecting two distinct primes commonly utilized in SNARKs and other cryptographic primitives, and enhancing the circuit to facilitate arithmetic operations of one field on another field. The objective is to optimize the circuit to support this capability efficiently.

Category 3: Special purpose ZKP protocols

Description: To improve the ZKP schemes one step further, the goal of this task is to further design special-purpose ZKP protocols with better prover time tailored for the computations described above. The participants can utilize techniques such as custom gates, lookup arguments or propose other special ZKP protocols.

Designated Tasks:

Designated Task 3.1: SHA-256
Designated Task 3.2: Keccak-256
Designated Task 3.3: BLS signature
Designated Task 3.4: ECDSA signature

Category 4: Circuit development in Halo2-ce

Description: Scroll will create support material for the ZKP Circuits tracks — this will walk builders through how to set up a dev environment, test circuit correctness, and assess performance for their hackathon project.

This dev environment will be using halo2-ce, which is an extension of the proof system used by the Ethereum Foundation's PSE team's zkevm project (which Scroll actively contributes to and uses in the Scroll network).

Scroll would love to see circuits built or further optimized that contribute to the ecosystem.

Specifically, Scroll researchers suggest that the following circuits be built or optimized in Halo2:
Designated Task 4.1: RIPEMD-160 hash function
Designated Task 4.2: blake2f hash function
Designated Task 4.3: SHA2-256 hash function

The Scroll team will provide a scaffold building environment and test cases for these functions.

Prize: Scroll will award the winning team with a $15,000 prize, on the condition that the circuit is open-source.

Category 5: Brief proofs of critical computations in blockchain applications using NEAR as an example

Description: We encourage community members to work with us in programming recursive proof systems for Near Protocol using Plonky2 and Rust frameworks. The goal is to design different aggregation and recursive circuits optimized for several practical cases using Near Protocol primitives and data. It is highly in demand on the market tradeoffs between size and proof generation speed, as well as on-chain verification.

Zpoken has experience in programming recursive proof systems for distributed blockchain ledgers using the Plonky 2 protocol and Rust frameworks. This knowledge and experience is useful for popularizing this direction among the students during hackathons and other competitive forms of education. We offer a series of simple tasks to learn the basics of the Rust programming language, the Plonky 2 protocol and recursive proof systems. As the tasks progress, they become more complicated and supplemented with new scenarios, which, together with the competitive form of training, stimulates the cognitive functions of students.

Brief overview of the task: NEAR Hackathon circuit category

Rewards: Team (or individual) who solved the practical task and submitted presentation first wins the prize of $10,000.

Review Mechanism

We will carefully review submissions with respect to the following criteria:

1. Correctness of the implementation.

We will provide reference implementations compiled automatically from existing libraries, such as circom, gnark and arkworks for the basic computations such as SHA256, Keccak-256, Blake-2, ECDSA and the Merkle tree path verification. The submissions have to demonstrate the correctness by providing the input and output pairs that are the same as the reference implementations.

For complicated tasks under recursive SNARKs, the participants should provide the documentation explaining the correctness of the circuit/R1CS.

2. Efficiency of the implementation.

For category 1, we will compare your schemes against the reference implementations generated from existing libraries in the same format of circuit or R1CS and report the improvement on the number of gates or R1CS constraints.

For category 2, we will compare the number of gates or R1CS constraints across the submissions.

For category 3, we will measure the prover time, proof size and verifier time of your schemes and compare them against established reference protocols such as Groth16 and Plonk, using automatically compiled circuits from existing libraries as baselines.

3. Quality of the documentation.

Participants are required to submit a comprehensive write-up detailing the optimizations implemented and their impact on the prover time, as well as any trade-offs in proof size or verifier time.

Award and Prize

The prize will be awarded to the submissions with the smallest circuit/R1CS size or the fastest prover running time for different tasks, with the requirement that they pass the test of correctness.

Prize: Up to $1000 for SHA-256, $1000 for keccak-256, $4000 for ECDSA, $4000 for Ed25519, in the zk framework of your choice. (Prize pool sponsored by Jump).